MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2017/2018

ECE3136 – JAVA TECHNOLOGY

(All Sections / Groups)

16 OCTOBER 2017
9:00 A.M – 11:00 A.M.
(2 Hours)

INSTRUCTIONS TO STUDENT

1. This examination paper consists of FIVE (5) pages including cover with THREE (3) questions only.
2. Answer ALL questions. All questions carry 25 marks and the distribution of the marks for each question is given.
3. This examination is OPEN-BOOK. Allowed material: Textbook and/or bound notes only.
4. Please print all your answers in the Answer Booklet provided.

Question 1

Class **Player** is shown in Figure Q1a.

```
public class Player
{
	private String name;
	private int faceValue;
	private final int MAX = 6;

	public Player (String n)
	{
		setName(n);
	}

	public void setName(String n)
	{
		name = n;
	}

	public String getName()
	{
		return name;
	}

	public int roll()
	{
		faceValue = (int)(Math.random() * MAX) + 1;
		return faceValue;
	}

	public String toString()
	{
		return name +  " (rolled: " + faceValue + ")";
	}
}
```

Figure Q1a: Class **Player**

Write an application class **Game** that plays a rolling dice game between user and computer. Instantiate 2 objects of class **Player**, one with the name "Computer" which is the default **Player** that the user plays with. Prompt user for a name and create another **Player** using that name.

[8 marks]

Prompt user for the number of rounds to play. Handle all exceptions that may occur in obtaining input.

[5 marks]

Play the game by rolling the dice for the number of rounds requested by user. At each round, compare the value rolled by user vs. computer. Display the status after each round, ie. whether the user rolled a higher number than computer (Win), a lower number than computer (Lose) or the same number (Draw). After the completion of the rounds, display the end result that tabulates the number of wins, losses and draws for the user. See Figure Q1b.

[12 marks]

Continued...

```
Your name : Claire
How many rounds to play?: 8
Computer (rolled: 6) is more than Claire (rolled: 2)
Claire (rolled: 3) is equal to Computer (rolled: 3)
Claire (rolled: 5) is more than Computer (rolled: 3)
Claire (rolled: 4) is equal to Computer (rolled: 4)
Computer (rolled: 4) is more than Claire (rolled: 3)
Computer (rolled: 5) is more than Claire (rolled: 3)
Claire (rolled: 4) is more than Computer (rolled: 3)
Computer (rolled: 5) is more than Claire (rolled: 4)

Claire: Wins:2  Losses: 4       Draws: 2
```

Figure Q1b: Sample run

Continued...

Question 2

Write a Java program that will display a window like this:

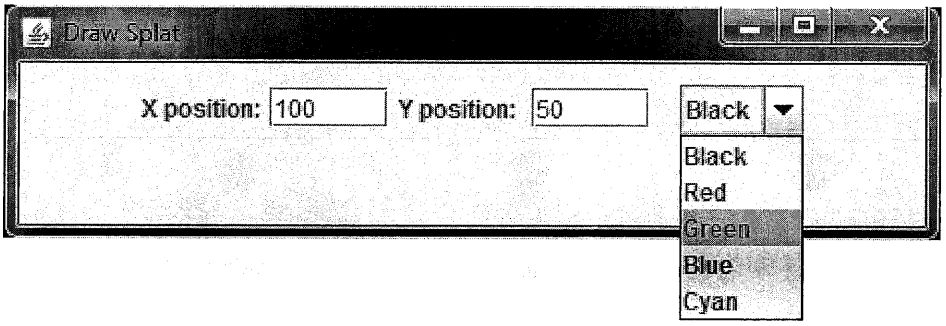

Figure Q2a: Main Window

When one of the combo box colours is selected, it will pop out a new frame and draw a filled circle (Splat) based on the position and colour.

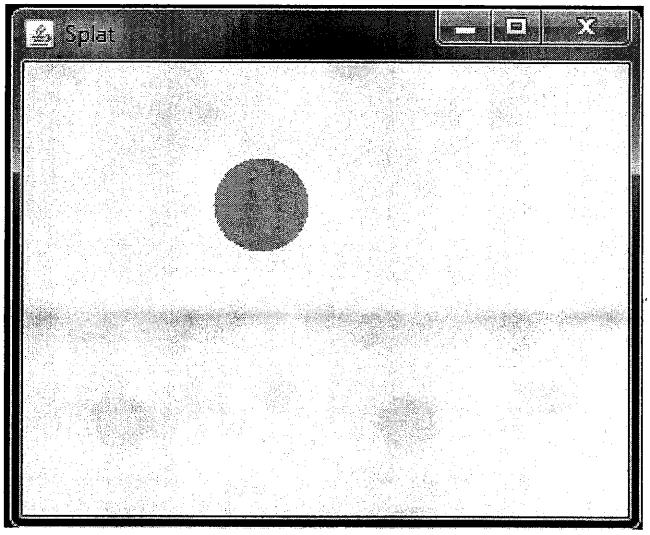

Figure Q2b: Pop-out Frame

In your program, write:

- A panel class with GUI elements (labels, textfields, combo box) as shown (See Figure Q2a). Attach a listener to the combo box.

  [11 marks]
- A main method that displays a frame containing the panel as in Figure Q2a.

  [3 marks]
- The combo box listener which pops out a new frame (Figure Q2b) with the Splat.

  [5 marks]
- A panel class on which the Splat (diameter 50 pixels) is drawn, based on the position and colour selected.

  [6 marks]

Note: The position is using the Java convention, that is, the y-position extends from top to bottom.

Continued...

Question 3

Create the class hierarchy shown in Figure Q3.

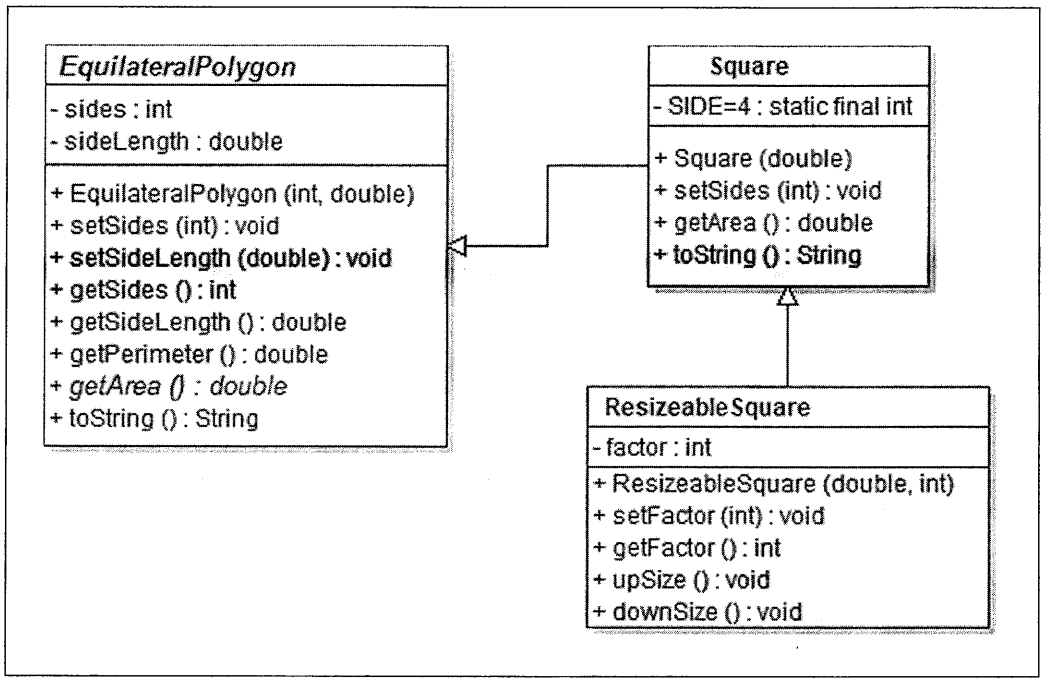
EquilateralPolygon
- sides : int
- sideLength : double
+ EquilateralPolygon (int, double)
+ setSides (int) : void
+ setSideLength (double) : void
+ getSides () : int
+ getSideLength () : double
+ getPerimeter () : double
+ getArea () : double
+ toString () : String
Square
- SIDE=4 : static final int
+ Square (double)
+ setSides (int) : void
+ getArea () : double
+ toString () : String
ResizeableSquare
- factor : int
+ ResizeableSquare (double, int)
+ setFactor (int) : void
+ getFactor () : int
+ upSize () : void
+ downSize () : void

Figure Q3

(a) Write an abstract class, **`EquilateralPolygon`**. The class has two instance variables, **`sides`** (the number of sides of the polygon object) and **`sideLength`** (the length of each side). Provide a constructor and all the methods listed in the **`EquilateralPolygon`** class diagram in Figure Q3.

[10 marks]

(b) Write a class, **`Square`**, which inherits from **`EquilateralPolygon`**. **`Square`** defines a static constant **`SIDE`** with value 4, to indicate all square objects have exactly 4 sides. Provide a constructor. Override the superclass abstract method and **`toString`** method. Override the method **`setSides`** to prevent instance variable **`sides`** from being changed to any values other than 4.

[8 marks]

(c) Write a class, **`ResizeableSquare`**, which inherits from **`Square`**. Provide a constructor, and accessor and mutator methods. Instance variable **`factor`** is used in resizing operations performed in methods **`upSize`** and **`downSize`**. Method **`upSize`** increases the side length of the square by **`factor`**, and **`downSize`** decreases it similarly.

[7 marks]

NOTE: DO NOT write the driver program.

End of Paper